Unsigned 16-bit Data

Brief recipe

Thu, Jan 10, 2008

Dealing with unsigned data words in the front end is done by keeping the internal data pool values offset by 32768, or `0x8000`. This allows mapping the unsigned range `0x0000–FFFF` to the signed range `0x8000–7FFF`. By using the linear formula for engineering units scaling, we can derive the full range of values.

```
eng = (raw/32768)*fs + off
```

If the engineering units are to be merely the unsigned integer value, use 32768 for both the fullscale and the offset. Given the signed raw values -32768 to +32767, the range of engineering units results will therefore range from 0–65535.

Assume that we have hardware that expects a 16-bit unsigned value when it is written, which is often the case for time delays. To access such unsigned hardware via memory address, use Data Access Table type `0x30`, which reads words of memory, toggling the sign bit before placing the readings into the data pool. Here is the entry format:

```
3000  chan  address32
0000  0000  step  count
```

The initial target address is given, along with step size and a count. This copies data words from the target address into the data pool readings, toggling the sign bit of each.

To set such hardware, use analog control type `0x18`. The 4-byte `ADESC` field is:

```
18  ad  dress
```

If the address is a 32-bit address, use the `MAP32` scheme to allow expansion to 32 bits. The `ad` byte indexes the `MAP32` table to get the most significant 16 bits of the address. The `dress` word provides the least significant 16 bits.

If the hardware is a 16-bit timer for which `0000` is an invalid value, use analog control type `0x1C` instead, as it will prevent a zero from being written, replacing it with `0001`.